# Beyond the Web: Excavating the Real World Via Mosaic

THE MERCURY PROJECT.

- Ken Goldberg, Assistant Professor, Computer Science
- Michael Mascha, Assistant Professor, Anthropology and
- Steven Gentner, M.S. Candidate, Computer Science
- Juergen Rossman, Graduate Student, University of Dortmund, Germany
- Nick Rothenberg, PhD Candidate, Visual Anthropology
- Carl Sutter, Senior Programmer/Analyst, Center for Scholarly Technology
- Jeff Wiegley, PhD Candidate, Computer Science

University of Southern California. Los Angeles, CA.

(To appear in the Second International WWW Conference, Chicago, IL, Oct 17-21, 1994.)

## Abstract

*This paper describes a Mosaic server that allows users to "leave the Web" and interact with the real world. An interdisciplinary team of anthropologists, computer scientists and electrical engineers collaborated on the project, desigining a system which consists of a robot arm fitted with a CCD camera and a pneumatic system. By clicking on an ISMAP control panel image, the operator of the robot directs the camera to move vertically or horizontally in order to obtain a desired position and image. The robot is located over a dry-earth surface allowing users to direct short bursts of compressed air onto the surface using the pneumatic system. Thus robot operators can "excavate" regions within the environment by positioning the arm, delivering a burst of air, and viewing the image of the newly cleared region. This paper describes the system in detail, addressing critical issues such as robot interface, security measures, user authentication, and interface design. We see this project as a feasibility study for a broad range of WWW applications.*

## Goals of the Project

WWW and Mosaic[1]-like servers provide a multi-media interface that spans all major platforms. Thousands of sites have been set up in the past year. Our goal with this project was to provide public access to a teleoperated robot, thus allowing users to reach beyond the digital boundaries of the WWW.

Such a system should be robust as it must operate 24 hours a day and it should be low in cost (we had an extremely limited budget). It is worth noting that the manufacturing industry uses the same criteria to evaluate robots for production. Thus our experience with RISC robotics (see below) proved helpful.

Our secondary goal was to create an evolving WWW site that would encourage repeat visits by users to collectively solve a puzzle. As of this writing we do not have sufficient data to report on the success of the "puzzle" component; therefore this paper focuses on the details of the implementation. We also speculate on how Mosaic might be used for other tele-operated applications.

# Related Work

The first "teleoperated robots" were developed over 30 years ago. The basic objective has always been to develop systems capable of working in inhospitable environments (such as radiation sites). Teleoperation began with very simple mock-ups in nuclear power plants [Mos], progressing to more versatile setups for teleoperation of robots in space [Miz]. Over the last 20 years, the development of intuitively operable teleoperation tools has continued to play an important role in the development of robotics in general. The basic objectives have remained the same, even though the methods and technical limitations have changed.

Today, sophisticated "Telerobot Operator Control Stations" [Kan] are equipped with stereoimage-displays, "force reflecting hand controllers" and comprehensive video graphics support. The development of teleoperation stations is currently being pushed further with the help of latest graphics workstations to provide so-called "telepresence." Modern telepresence systems, considered to be pushing the frontier of research in this field, are defined as follows [Aki]: "At the worksite, the manipulator has the dexterity to allow the operator to perform normal human functions. At the control station, the operator receives sufficient quantity and quality of sensory feedback to provide a feeling of actual presence at theworksite."

The Mercury Project does not achieve this level of telepresence but provides a limited level of teleoperation. One of our goals was to provide "teleoperation for the masses." Instead of developing a highly sophisticated, multi-million-dollar testbed, we opted for a simple and reliable end-effector on a commercial robot. Combined with an intuitively operable man-machine-interface, the system gives all WWW users access to teleoperation.

In the Discussion section, we describe a number of other WWW sites that offer interactive capabilities.

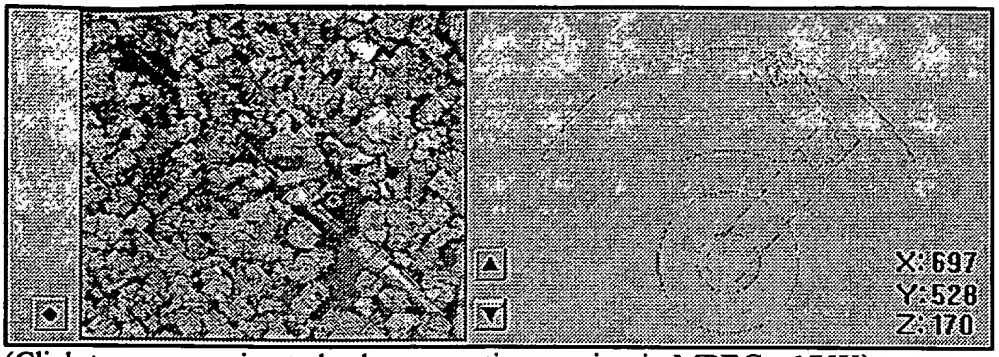
# User Interface and Environment Design

The interface design for the system was challenging due to the limitations of the HTML/HTTP environment, as well as network traffic considerations. An effective system was created within such limitations by carefully designing the physical environment for the robot, and by fine-tuning the user-machine interface. For example, the initial idea of a live video feed from the camera was dropped in order to maintain compatibility with all visual clients on the Web. (Although we could have implemented some custom clients [2], we decided to stay within the limits of HTML/HTTP to reach as large a user base as possible, making this a truly global system.) In addition, initial simulations using a robot fitted with grippers (simulated in VIRTUS WALKTHROUGH) revealed a high degree of complexity in control functions [3], not suitable for the anticipated 5-10 seconds per frame page loading time, a 2D Mosaic window and a naive/untrained user.

The team chose instead to use a simple environment which would allow relatively easy control of the robot. Here the analogy taken from real world archaeology - using a dry-earth environment and compressed air bursts - allowed us to simplify the robot control dramatically. Thus users could be quickly trained in the operation of the system, through a simple "OperatorÕs Orientation" and a "Level 1 Clearance Test."

Even with a simplified system, users are still able to choose between fine and gross movements of the arm. Fine pitch movements are executed by clicking in the camera image, with the robot moving to center the arm over the X,Y coordinates of the click-point. Crude navigation is provided by clicking on a

schematic picture of the robot and it's workspace, with the robot moving to center the arm over the click-point. Two buttons allow navigation in the Z axis (between "up" and "down" positions), with a button to blow air only active when at the Z=O (i.e., "down") position.




(Click to see an animated robot operation session in MPEG - 175K)

Other features of the system were designed to balance functionality with user needs. All HTML documents sent to the clients are carefully designed to minimize network traffic in order to get a high refresh rate. For example, control panel functions are clearly distinguished from text-based information documents. The ÒOperator's LogÓ was implemented to create a forum for collaborative efforts to solve the puzzle/problem regarding the underlying logic which links the artifacts. (The ÒOperator's LogÓ is readable throughout the system but only writeable after completing an operating session.) A second entry path was also created to the system, which provides a "back-story" explaining the project while also hinting at possible "real world" uses of the system.

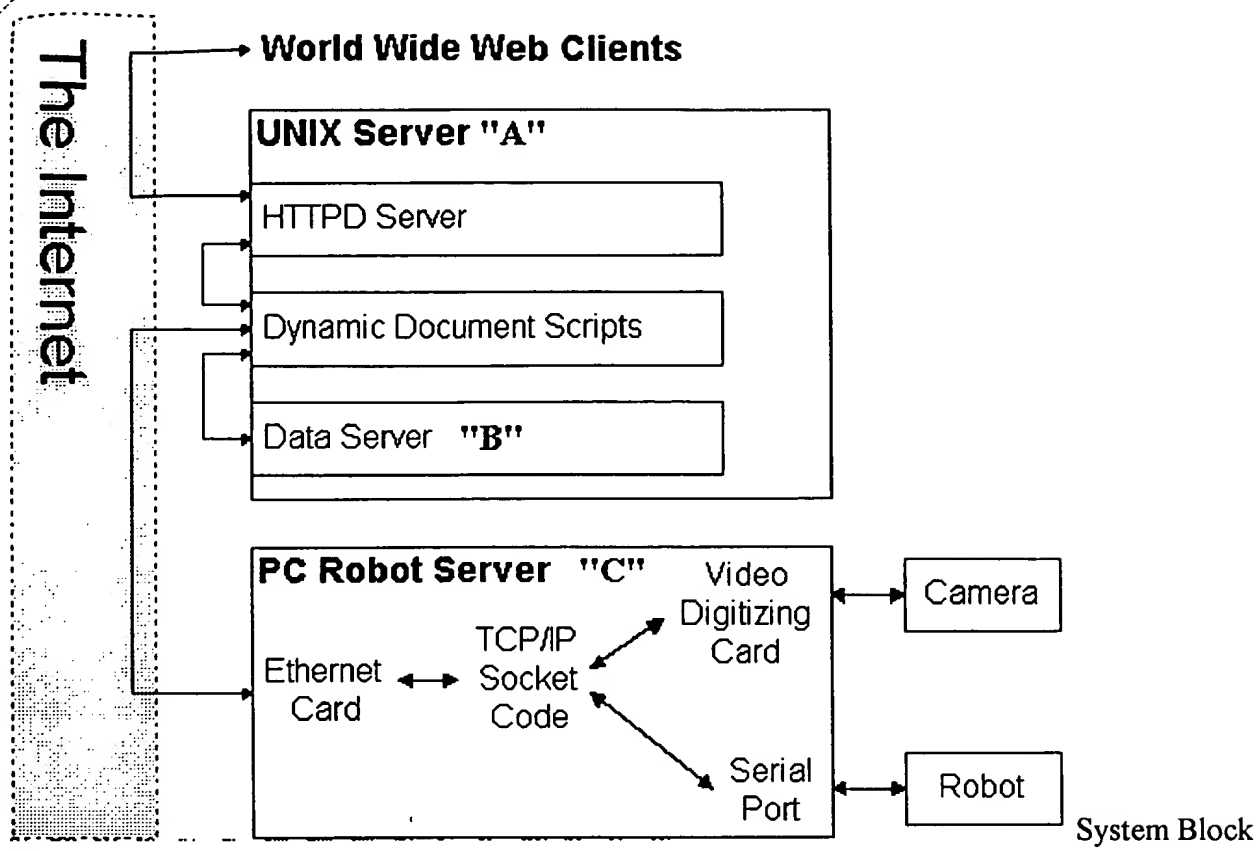
# Access to the Robot

Most of the HTML documents seen by the user on our site are generated by a script running on the WWW server. Using a random token scheme described below, the system tracks each user as he or she proceeds through the interface and generates appropriate HTML documents. This allows the system to discriminate between "observers" and "operators" so that it presents only accessible options to each.

To operate the robot, the user must read the information on how to use the control panel, and then complete a level-1 clearance test to get a password. Since only one person can operate the robot at a time, the system maintains a queue of pending operators. A typical user will enter his/her password, and then add him/herself to the queue. Each time update button is clicked, the system updates the queue and returns a current status page. When the user's turn arrives, the screen returned is the live operators' control screen.

# System Architecture

Below is a Block Diagram for the system. We start with an overview that necessarily glosses over many interesting details.




System Block Diagram

At one end are WWW clients from around the world; at the other end is a robot arm combined with a camera. The robot and camera provide an updated image of the environment, which is combined with a schematic of the robot arm/workspace and control buttons to produce the final GIF image that is send to users.

At any given time there may be dozens of clients interacting with the system. Since there can only be one Operator at a time, one challenge is to keep track of which client is the operator.

The Mercury system is comprised of three communicating servers. The first, call it A, is a standard Mosaic server (NCSA httpd v.1.3, currently running on a Sun SPARCserver 1000, with SunOS Release 5.3. When the RTE Site is requested by an observer, the most recent image, which is stored on server A, is simply returned.

The database of registered users is handled by another server, call it B. In our case, Server B runs on the same machine as server A. The database server is custom programmed for this project, but performs fairly standard database functions.

When a client request comes in, Server A communicates with server B. If that client is an Operator, Server A must then communicate with a third server, call it C, that controls the robot. Server C runs on a Pentium-based PC and communicates with servers A and B via the Internet. Server A decodes the ISMAP X&Y mouse coordinates, and sends them to server C.

On server C, a custom program decodes these coordinates into a robot command and verifies that the command is legal, e.g., within the robot workspace. If it is, this command is then converted into a robot command format which is sent to the robot over a serial line. Once the robot move is completed server C

uses the CCD camera to capture a stable 8 bit 192x165 image of the workspace.

Using a simple set of equations for inverse kinematics server C then generates a schematic view of the robot in its new configuration. This schematic is combined with the camera image, and the up, down, and air control buttons to form a new composite image. Server C then compresses this image into GIF format and returns it to Server A, which updates the most recent image and returns it to the Operator client.

# Subsystems

## Random Tokens for Cache Avoidance and User Tracking

Following some complex and unwieldy tests, we implemented a random token scheme for tracking users as they use the system. Each time a URL is returned, a large random number is added to the path (which the NCSA HTTPD 1.3 server splits into the PATH_INFO environment variable). This "token" serves two purposes:

The first is to prevent the WWW client from caching the robot view. When a document is requested a second time during a session, it is much faster to swap in a local copy of the document rather than going back over the net to retrieve it a second time. Most implementations of Mosaic support such caching at various memory levels. However in our case we want to repeatedly retrieve the URL containing the robot image because it is updated continuously. In brief, we DON'T WANT users to cache this url. The random token makes each request look different and tricks the client into retrieving a fresh version of the document.

The second use for the token is to identify Operators. When an operator logs in with a correct password, the system begins tracking him/her as he/she moves from viewing the robot to being on the operators' queue to operating the robot. Since the same script is used for all views, the token allows the system to customize the result for every user depending on his/her position in the system.

## Scripts

The robot view screen is controlled for the most part by one script at the HTTPD server. Each call to the main script has a token attached to the URL. The token is decoded by the WWW server, and placed in the PATH_INFO environment variable. The main script then checks the token with the database server to determine the status of the user. Each check of the database generates a system update to keep the queue moving. The user's status is used to generate the custom system status page.

The robot image itself is only changed by the operator when he or she makes a move. Each image is date and time stamped, so WWW clients that cache the image will only retrieve the image when it changes (since its filename will be different, due to a different time stamp).

Due to the client-server architecture of the World Wide Web HTTPD protocol, The robot system (server) has no way to contact the client except at the client's prompting. From the user's point of view, once he or she gets the robot view screen, there is no way for the server to keep sending updates automatically as the robot is moved by the operator. The screen updates must be driven by the user. Since he user must trigger each update, we wanted to provide a button for doing so, since each web client handles reloading the page differently. Some sites have a "reload" hypertext link to the same page, but his doesn't work for any client that caches pages. If a page is being viewed, hitting reload will just re-display the page from the cache, thus not obtaining a new view from the system. Asking the user to

disable his/her cache is also problematic, since not all clients allow this option.

One attempt was made to use a mini-form, since the submit button always calls a script and is not cached. that scheme was eventually dropped, since passing registered user identification information to the server via hidden fields only worked on some clients. Using the random token allows for an elegant interface.

Since the robot can only be controlled by one person at a time, a registration scheme was implemented to allow the server to track operators as they move on to the waiting queue and progress to controlling the robot. Since the server only knows the IP address of each user, some user information had to be incorporated into the HTML robot view document itself for re-transmission to the system when the user hits "reload." There are various techniques used by many sophisticated web systems to accomplish user identification between document requests, but we found some problems in many of the standard solutions. In the end, the random token served excellently as a means of identifying registered users.

A preliminary attempt was made to use a small form to identify the user. Hidden fields could hold the user id, but once again, many clients do not implement the hidden field attributes so the interface is cluttered by unnecessary fields. Putting the user's id information into the ACTION field of the form tag is also client dependent. Unfortunately, some clients strip that data before adding the encoded field information.

Since random tokens were already being passed with each update, the system was extended to track the tokens of each registered user. Each time the script is called, the token is exchanged for a new one, and the database is updated with the new token for registered users. One side effect is that the user can not use the client re-load button, since this will not use the new URL (it is embedded in the update HREF).

## The Data Server

The data server ("B") is a custom Perl script that handles all of the database work for the project. It continuously runs as a TCP/IP listener, waiting for database transactions from the other system scripts. The data server runs as a single process, handling requests serially to maintain internal data integrity. Typically, transactions are very short, since the data is kept in main memory. The data server could be replaced by an off-the-shelf transaction based DB system in the future. A time-out is set to close the connection if there is too much time elapsed between commands. This was implemented because some WWW clients would crash in the middle of a document request, leaving the system waiting for the connection to be closed.

## Internal Network Interface

The networking functionality required by the project was defined by two factors. On one hand, the camera that we purchased required a PC-based platform running an MicroSoft DOS or compatible operating system to run on Server C. On the other hand, the expected load of client requests required a machine capable of more heavy networking duties such as a Sun workstation (Server A). Currently Server A is located across campus from server C.

These servers are connected via Ethernet. Each machine has its own IP address and resides in the usc.edu domain. Communication is achieved using a socket connection between the two machines. The implementation on Server A was done using the standard BSD socket functions provided with the SunOS 4.1 operating system and Perl. On Server C we used a publicly available socket package called Waterloo TCP and Borland C. The Waterloo TCP package was obtained from the ftp site dorm.rutgers.edu in the

file /pub/msdos/wattcp/wattcp.zip.

With this software Server A can request a socket connection to Server C to establish a connection. The first step in obtaining a new image is for Server A to write a command consisting of thirty bytes which encodes the (xy) coordinates of the ISMAP event. After Server C completes the moves and generates the new image, it writes the size of the new image to server A so that server A knows exactly how many bytes to expect. Server C then proceeds to write the entire image to the socket and waits for the socket to close to insure deliver of the data. Once server A has read all the specified bytes it closes the socket. Server C is now ready and waiting for another socket connection. Server A is free to continue processing the Mosaic actions of the current users.

Current throughput is approximately 20 Kbytes/second, which is poor compared to the 0.5 megabyte per second rate that can be achieved between two Sun workstations in close proximity on the campus network. At this time we feel that the delays are being imposed by the MS-DOS operating system because of it's inability to support networking operations and its lack of multitasking abilities, which necessitates busy waiting cycles in the PC software to obtain concurrence between the robotic/camera operations and the networking duties.

Our low data rate is somewhat tolerable because the time for communication between Servers A and C is small compared with Internet delays between clients and server A. One way to speed communication would be to use different methods of image compression such as JPEG to reduce the size of the image. However this may introduce latency due to encoding.

## The IBM Robot and Server "C"

The robot we're using is an IBM SR5427 SCARA arm, built around 1980.

SCARA stands for "Selective Compliance Assembly Robot Arm" . Robots with SCARA kinematics are common in industrial assembly for "pick-and-place" operations because they are fast, accurate and have a large 2.5D workspace. However, the SCARA arm can only rotate its gripper about the vertical (Z) axis. We selected this robot over other robots in our lab due to it's excellent durability, large workspace, and because it was gathering dust in the Robot Education Lab.

The IBM SCARA robot is controlled through a 4800 baud serial port by a custom written C library constructed with reference from IBM's BASIC library distributed along with the robot. The commands sent by the library are simple instructions consisting of instruction id, length, data and checksum. The data length and content varies depending on instruction id. The IEEE floating point format is used to represent the necessary data. This command string is then sent over the serial line to the robot to issue the command.

Unfortunately IBM no longer supports this robot and we were forced to read two antiquated BASIC programs and monitor their serial line transmissions to decipher the protocols needed for serial control of the robot. The robot accepts XYZ and Theta commands using IEEE format and checksums. Server C now runs on a Pentium based PC with all custom code written in Borland C.

The first step was implementing a local graphical user interface to control robot movements and monitor subsequent functions such as network flow. We chose two views of the workspace: a global schematic view for coarse motions, and a local camera view for fine motions. Note that a click on the camera image requires a different relative move if the camera is in the up or down position. To handle it, we

implemented an empirical calibration program.

The major difficulty in implementing Server C was to schedule response to the network, the local mouse, and the image capture board. At first we discussed a multi-tasking environment but, upon further study, we realized we could achieve this cooperation within a single DOS task. Another problem, inherent to DOS based applications, is memory management. This complication was solved by careful usage of memory and by utilizing the screen itself as a memory buffer. This careful usage of memory enabled the custom written GIF encoder to use more memory which, combined with an appropriate hash function, sped the GIF encoding process up to a few microseconds.

In future versions of Mercury, we plan to incorporate a more sophisticated PC-based robot simulation system based on COSIMIR [Fre] from the University of Dortmund.

## Camera

We are using an EDC 1000 digital CCD camera from Electrim Inc. This camera was chosen based on size and cost. Image data is sent from the camera back through a serial line into a video capture card. The picture captured is always 192 by 165 pixels with 256 shades of gray. The image size and gray shades are fixed. Focus and contrast are manually adjusted. Exposure time can be changed by software to range between 1/200 th to 1/64 th of a second. 1/150th exposure was used to reduced light streaking that the camera is prone to.

Although the robot's control system quickly dampens oscillation about the destination point, dynamic effects can cause image blur. Two solutions were implemented. First the robot was slowed down enough as to reduce some of the vibration but not to hinder the robot access speed considerably. Second, once the robot responds positively to an issued command, the camera captures two pictures each at 1/64 th of a second. These two images are compared to determine a factor of similarity. If this factor is below some set value the image is presumed to be stable, otherwise subsequent pictures are taken until the image pair is determined to be stable. More then 5 trials results in a time-out in which case the most current image is used and the program continues. This image comparison procedure reduces movement streaks seen in pictures of moving objects.

Lighting the workspace has been problematic. The work space is primarily luminated by standard florescent ceiling fixtures and augmented by two additional florescent lamps to reduce shadows and raise the overall ambient light levels. We tested a contrast enhancement routine to normalize the lighting of each image captured from the camera. This increased the visual aesthetics of the image but subjected it to drastic light and dark changes as the robot moved onto different objects with different light reflecting qualities. In response, a global lighting adjustment was implemented but found to reduce certain areas to unacceptable darkness. Certainly a better lighting system is required.

Due to the manual focus adjustment of the camera, the focus adjustment could not be changed between the up and down position of the camera. This resulted in a compromise focus adjustment that is not perfect for the up or down position of the robot arm, but accepatable in both positions.

To decrease compressed image size and thus increase network transfer rate the image is reduced from 256 to 64 gray scales since most systems available can only display 256 colors or 64 shades of gray. Thus the gray scale reduction did not reduce image quality but reduced compressed image size by about 10K.

## Robustness and Soft Resets

All robot motions are monitored by Server C. Each command sent to the robot is verified to be within the robot's workspace. Acknowledgments from the robot are monitored to detect errors. When an error is detected, Server C automatically resets the robot controller, recalibrates, and returns the robot to its previous position.

# Performance

## History and Statistics

Daily statistics are available and may be correlated with project milestones. As of the writing of this paper, the system has been online for over 4 weeks and there are approximately 100 users per day. There is also a list of all hosts that have visited the system. As of this writing the system has been visted by hosts from all of the major continents except the polar caps.

## Refresh Rates via Ethernet

System response time seems to be mostly dependent on network link speeds. Locally, we get screen refreshes at rates of 5-10 seconds per page. Similar response times have been reported from Europe. Obviously, a slow local link or SLIP connection will drastically affect the update speed, since the robot control image is essential to the system. Updates are also strongly affected by the speed of the WWW client application.

## Uptime

The system is designed for 24 hour use. The WWW server scripts are generally modified, tested and then loaded into the running system. Background programs monitor the system and notify the team members if there are problems.

## Operators' Logs

When an operator has finished driving, he or she is prompted to make a textual entry into an "Operator's log". The Operator's log provides an ongoing forum for discussion of the system and record of artifacts discovered in the sand.

For example, several skeptics have claimed that the system is an elaborate hoax where all images are taken from a prestored library (much like the celebrated Apollo Moonwalk hoax of 25 years ago). We have had encouraging comments from the robotics community, including several researchers at NASA.

# Discussion and Future Applications:

This project is an initial step in an ongoing educational and research project at the University of Southern California. It brings together faculty and students of different backgrounds to collaborate in the design and implementation of a networked system that combines robotics with archaeology and interactive art.

This system exemplifies RISC Robotics, which advocates Reduced Intricacy in Sensing and Control. The SCARA-type robot requires only 4 axes, is relatively inexpensive and robust, and it is easy to avoid singularities. The end effector we've used here is also about the minimum. For more on RISC as applied

to industrial robotics, please see RISC for Industrial Robots: Recent Results and Open Problems, (with J. Canny), 1994 IEEE Conference on Robotics and Automation.

We see the project leading in several directions. For Mosaic and the WWW, the required interface design prompted new developments related to several issues, including user authentication, user queuing and interface security (as discussed above).

For this project we chose a very simple application. The server can be extended to a variety of platforms that permit remote inspection and manipulation of objects -- for example, providing unique and unedited access and views of priceless and otherwise inaccessible resources (a Grecian urn, a Gutenberg Bible, etc.), thus providing an alternative to pre-stored libraries which are limited in terms of perspective, depth of resolution, etc. (Cite Recent NY Times article on the Metropolitan Museum of Art).

Further extensions for this project might include: the robot could be placed out in the field, in a remote anthropological site or on the moon; the camera could be replaced with a scanning electron microscope; or the remote operator could be a doctor examining a patient or a specialist performing remote inspection or manufacturing. All of these areas also have significant implications for education, as they present the opportunity for virtual "field trips" to a live site while permitting remote manipulation from the classroom.

Anthropologists have conventionally recorded the diverse cultural heritage of humankind by means of varied media: written text, graphics, film, sound and still images. The advantage of a system like the one described in this paper lies in the fact that you do not have to rely on prerecorded media. It enables the user to view and possibly record her or his own "slice of reality". We see the Web as a perfect medium for updating pre-recorded media as described in [Mas] Interactive Education: Transitioning CD-ROMs to the WEB, a paper presented at the First WWW Conference, Geneva 94. Furthermore, we now have the possibility to combine updateable prerecorded media of all sorts with live recordings and live remote interactions. The possibilites of a system that combines global access to up-datable prerecorded media and combines it with the possibility of live remote interactions are just beginning to unfold, and are a central focus of interest for the anthropologists from the E-LAB involved in this project.

In view of other interactie WWW sites, we propose a three-tiered system describing interactively on the WWW. Under Level I, interaction is solely between digital information stored on computers or created by scripts running on such machines, and connected or communicating with the WWW and Mosaic clients. In Level II sites, the clients are able to observe the "real world" by means of a camera observing and digitizing visual and, hopefully soon, audio-visual information. The camera acts as an "eye" for the Web, providing multiple "real world images" from a global theater. A number of Web sites fall into this category, such as the Coffee Pot and the Fishtank sites. All have the same characteristic of passively observing the real world. We also know of one restricted site that allows the user to alter the user's point of view (see LEGO pan and tilt site.)

The Mercury Project introduces a third level. Level III sites reach beyond the digital domain to allow users to alter a remote physical environment. We envision this project as a first glimpse into the possibilities available at Level III. We might also speculate about other levels, which might allow remote users to control a mobile robot and thus "tele-ambulate".

# Footnotes

[1]

To simplify we mention only Mosaic as a WWW client but we are aware of the fact that there are different WWW clients similar to Mosaic, e.g. MacWeb, Cello, etc.

[2]

MBONE broadcasting, REFERENCE to MIT LIVE VIDEO SITE [diversion - possible fixes to client refresh problem to show we know about the X stuff etc.] There are two possible fixes to this problem, One is to release specially modified clients that set up a two-way communication, the second is to use some other software to display the current system on the user's client workstation. Since many clients are used to view the WWW, making modifications would be difficult, especially since they are being updated all the time. Even if source code could be obtained for every major client, changes would have to made to every release of all these be on each release of these applications, The second possibility is to write a separate program to run on the clients' workstation. The problem here is to write a robots client that can be released for enough platforms to be useful, Since this would be an esoteric piece of the system, it is not likely that other sites would customize the software for different systems like is done for the major systems. One technique is to use the X windows protocol to display a client application on the users workstation running an X server. (weather, movies) We felt that this would be a limited audience, however. It also may compromise security from the user's point of view. Both these approaches may be attempted in version 2.0 of the system to allow more enhanced use of the system for some users. The HTTPD protocol could be extended to allow these sort of connections, though - maybe we need a new protocol for passing media only back that doesn't have all the hooks into system calls like X Windows and Display PostScript.

[3]

3D control of a robot needs: 3 dimensions of spatial movement, 3 dimensions of orientation and 1 to 3 dimensions of gripper control.

# Acknowledgments


- Rick Lacy, Mark Brown, and the Center for Scholarly Technology,
- Depts. of CS and Anthro,
- Prof. Peter Danzig (CS)
- The alpha and beta testers
- The Laika Foundation
- The Los Angeles Museum of Miniatures


# References


[Aki]

D.L. Akin, M. Minsky, e.a.:"Space Applications of automation, robotics, and machine intelligence systems (ARAMIS)",Phase II, Vol. 3, M.I.T. Contract NASA 8-34381, NASA Marshall Space Flight Center.

[Kan]

E.Kan, J.Tower e.a.:"The JPL Telerobot Operator Control Station: Part 1 - Hardware", Proceedings of the NASA Conference on Space Telerobotics, 1989.

[Mak]

B.J. Makinson:"Research and Development Prototype for Machine Augmentation of Human Strength and Endurance", Report S-71-1065, General Electric Company, Schenectady, NY, 1971

[Mos]

R.S. Mosher: "Force Reflecting Electrohydraulic Servomanipulator", Electro-Technology, pp. 138, 1960

[Miz] N.J. Mizen, "Preliminary Design for the Shoulders and Arms of a Powered Exosceletal Structure", Cornell Aeronautical Laboratory Report VO-1692-V-4, 1965

[Fre] Freund, E.; Rossmann, J.; Uthoff, J.; van der Valk, U. : "Towards realistic Simulation of Industrial Robots", Proceedings of the IEEE/RSJ/GI Intelligent Robots and Systems, IROS, Muenchen, 1994